

PATENT
10830.0097.NPUS00

APPLICATION FOR UNITED STATES LETTERS PATENT

for

**MULTI-PROTOCOL SHARABLE VIRTUAL STORAGE OBJECTS
MANAGEMENT AND CONTROL**

By

Rui Liang

Jiannan Zheng

Mark K. Ku

Xiaoye Jiang

Sorin Faibish

Express Mail Mailing Label No. EV318623769US

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to data processing networks including multiple clients and servers such as data storage systems and network file servers. The present invention specifically relates to the sharing of data storage objects between clients and servers using different interfaces, protocols, or operating systems.

2. Description of the Related Art

Network data storage is most economically provided by an array of low-cost disk drives integrated with a large semiconductor cache memory. A number of data mover computers are used to interface the cached disk array to the network. The data mover computers perform file locking and file metadata management and mapping of the network files to logical block addresses of storage in the cached disk array, and move data between network clients and storage in the cached disk array. Typically the logical block addresses of storage are subdivided into logical volumes. Each logical volume is mapped to the physical storage using a respective striping and redundancy scheme. The data mover computers typically use the Network File System (NFS) protocol to receive file access commands from UNIX and Linux clients, and the Common Internet File System (CIFS) protocol to receive file access commands from MicroSoft (MS) Windows clients.

More recently there has been a dramatic increase in various ways of networking clients to storage and protocols for client access to storage. These networking options include a Storage Area Network (SAN) providing a dedicated network for clients to access storage devices directly via Fibre-Channel, and Network Attached Storage (NAS)

1 for clients to access storage over a Transmission Control Protocol (TCP) and Internet
2 Protocol (IP) based network. In addition to the high-level file-access protocols such as
3 NFS and CIFS, the various networking options may use lower-level protocols such as the
4 Small Computer System Interface (SCSI), the Fibre-Channel protocol, and SCSI over IP
5 (iSCSI). However, most network facilities for data sharing and protection are based on
6 file access protocols, and therefore the use of lower-level protocols in lieu of file access
7 protocols for access to network storage may limit the available options for data sharing
8 and protection.

9 SUMMARY OF THE INVENTION

10 In accordance with one aspect, the invention provides a method of access to a
11 storage object in a file server. The file server and a client are included in a data
12 processing network. The method includes the client using a block level access protocol
13 over the network to access the storage object; and the file server accessing the storage
14 object by accessing a file containing data of the storage object.

15 In accordance with another aspect, the invention provides a method of access to a
16 virtual direct access storage device in the file server. The file server and a client are
17 included in a data processing network. Attributes and data of the virtual direct access
18 storage device are stored in at least one file in the file server. The method includes the
19 client using a block level access protocol over the network to access the virtual direct
20 access storage device in the file server. The file server responds to commands in
21 accordance with the block level access protocol for access to the virtual direct access
22 storage device by accessing the attributes and data of the virtual direct access storage
23 device. The method further includes the file server providing access over the network to

1 the virtual block storage device in accordance with a file access protocol by accessing the
2 at least one file in the file server.

3 In accordance with yet another aspect, the invention provides a network file
4 server. The network file server includes data storage, an interface for coupling the data
5 storage to a data network; and at least one processor programmed for permitting clients in
6 the data network to access the data storage in accordance with a plurality of access
7 protocols. The data storage contains at least one file for storing file attributes and
8 metadata defining a virtual direct access storage device and for storing data of the virtual
9 direct access storage device. The access protocols include at least one block level access
10 protocol for access to the virtual direct access storage device by accessing the metadata
11 and data of the virtual direct access storage device. The access protocols also include at
12 least one file access protocol for accessing the at least one file.

13 In accordance with a final aspect, the invention provides a network file server.
14 The network file server includes data storage, an interface for coupling the data storage to
15 an IP data network, and at least one processor programmed for permitting clients in the
16 data network to access the data storage in accordance with a plurality of access protocols.
17 The data storage contains at least one file for storing file attributes and metadata defining
18 a virtual SCSI direct access storage device and for storing data of the virtual direct access
19 storage device. The access protocols include a SCSI block level access protocol for
20 client access to the virtual SCSI direct access storage device over the IP network by
21 accessing the metadata and data of the virtual direct access storage device. The access
22 protocols further include at least one file access protocol for accessing said at least one
23 file. The network file server further includes a facility for remote replication of the at

1 least one file over the IP network concurrent with client write access to the virtual SCSI
2 direct access device over the IP network using the SCSI block level access protocol. The
3 remote replication facility may use a snapshot copy facility for replication by transmitting
4 read-only versions (i.e., snapshots) of the at least one file over the IP network.

6 BRIEF DESCRIPTION OF THE DRAWINGS

7 Other objects and advantages of the invention will become apparent upon reading
8 the following detailed description with reference to the accompanying drawings wherein:

9 FIG. 1 is a block diagram of a data processing system including multiple clients
10 and network file servers;

11 FIG. 2 is a block diagram showing in greater detail one of the clients and one of
12 the network file servers in the data processing system of FIG. 1;

13 FIG. 3 is a block diagram of a command in accordance with the Small Computer
14 System Interface (SCSI) protocol;

15 FIG. 4 is a block diagram of a SCSI Command Descriptor Block (CDB) in the
16 SCSI command of FIG. 3;

17 FIG. 5 is a block diagram of a storage object container file;

18 FIG. 6 is a flow chart of command execution by a SCSI termination module in the
19 data mover of FIG. 2;

20 FIG. 7 is a more detailed block diagram of the client in FIG. 2;

21 FIG. 8 is a more detailed block diagram of the data mover in FIG. 2;

22 FIG. 9 is a block diagram of a data packet for a Network Block Services (NBS)
23 protocol;

1 FIG. 10 is a table of client opcodes for the NBS protocol of FIG. 9;
2 FIG. 11 is a table of server opcodes for the NBS protocol of FIG. 11;
3 FIG. 12 is a block diagram showing control flow through the client and server of
4 FIGS. 7 and 8 for processing storage object container file snapshot and replication
5 requests from a system administrator;
6 FIGS. 13 and 14 comprise a flow chart of operation of the virtual block device
7 manager in FIG. 12 for processing a snapshot or replication request from the system
8 administrator; and
9 FIG. 15 shows a file system for containing a data storage object.
10 While the invention is susceptible to various modifications and alternative forms,
11 specific embodiments thereof have been shown in the drawings and will be described in
12 detail. It should be understood, however, that it is not intended to limit the invention to
13 the particular forms shown, but on the contrary, the intention is to cover all
14 modifications, equivalents, and alternatives falling within the scope of the invention as
15 defined by the appended claims.

16

17 **DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS**

18 FIG. 1 shows an IP network 20 including multiple network file servers 21, 22, and
19 multiple clients 23, 24, 25. The clients and network file servers, for example, can be
20 distributed world wide and linked via the Internet. Each of the network file servers 21,
21 22, for example, has multiple data mover computers 26, 27, 28, 32, 33, 34, for moving
22 data between the IP network 20 and the cached disk arrays 29, 35 respectively. Each of
23 the network file servers 21, 22 also has a control station 30, 36 connected via a dedicated

1 dual-redundant data link 31, 37 among the data movers for configuring the data movers
2 and the cached disk array 29, 35. Further details regarding the network file servers 21, 22
3 are found in Vahalia et al., U.S. Patent 5,893,140, incorporated herein by reference.

4 FIG. 2 shows software modules in the client 23 and the data mover 26 introduced
5 in FIG. 1. The data mover 26 has a Network File System (NFS) module 41 for
6 supporting communication among the clients and data movers of FIG. 1 over the IP
7 network 20 using the NFS file access protocol, and a Common Internet File System
8 (CIFS) module 42 for supporting communication over the IP network using the CIFS file
9 access protocol. The NFS module 41 and the CIFS module 42 are layered over a
10 Common File System (CFS) module 43, and the CFS module is layered over a Universal
11 File System (UxFS) module 44. The UxFS module supports a UNIX-based file system,
12 and the CFS module 43 provides higher-level functions common to NFS and CIFS.

13 The UxFS module accesses data organized into logical volumes defined by a
14 module 45. Each logical volume maps to contiguous logical storage addresses in the
15 cached disk array 29. The module 45 is layered over a SCSI driver 46 and a Fibre-
16 channel protocol (FCP) driver 47. The data mover 26 sends storage access requests
17 through a host bus adapter 48 using the Fibre-Channel protocol, the SCSI protocol, or the
18 iSCSI protocol, depending on the physical link between the data mover 26 and the cached
19 disk array 29.

20 As introduced above, some clients may desire to use lower-level protocols such as
21 the Small Computer System Interface (SCSI), the Fibre-Channel protocol, and SCSI over
22 IP (iSCSI) in order to access network storage. One environment where this is desirable is
23 a Microsoft Exchange platform. In this environment, a Microsoft Exchange server, or a

1 server for a database such as an Oracle or SQL database, typically stores its database
2 component files and tables such as storage groups, and transaction logs to one or more
3 block devices. It is desired to replace these block devices with remote block devices in a
4 network file server, and to provide disaster protection by replicating the database files
5 and transaction logs to a geographically remote network file server and taking read-only
6 copies or snapshots of the database and logs, for backup to tape.

7 For the data processing network in FIG. 2, for example, the client may use iSCSI
8 over the IP network 20. In this example, the software modules in the client 23 include
9 application programs 51 layered over an operating system 52. The operating system
10 manages one or more file systems 53. To access the network storage, the file system
11 routines invoke a SCSI device driver 54, which issues SCSI commands to an iSCSI
12 initiator 55. The iSCSI initiator inserts the SCSI commands into a TCP connection
13 established by a TCP/IP module 56. The TCP/IP module 56 establishes the TCP
14 connection with the data mover 26, and packages the SCSI commands in IP data packets.
15 A network interface card 57 transmits the IP data packets over the IP network 20 to the
16 data mover 26.

17 A network interface card 61 in the data mover 26 receives the IP data packets
18 from the IP network 20. A TCP/IP module 62 decodes data from the IP data packets for
19 the TCP connection and sends it to an iSCSI target software driver module 63. The
20 iSCSI target module 63 decodes the SCSI commands from the data, and sends the SCSI
21 commands to a SCSI termination 64. The SCSI termination is a software module that
22 functions much like a controller in a SCSI disk drive, but it interprets a storage object 65
23 that defines a logical disk drive. The SCSI termination presents one or more virtual

1 LUNs to the iSCSI target 63. A virtual LUN is built on top of the storage object 65, and
2 it emulates a physical SCSI device by implementing SCSI primary commands (SPC-3)
3 and SCSI block commands (SBC-2).

4 Instead of reading or writing data directly to a physical disk drive, the SCSI
5 termination 64 reads or writes to a data storage area of the storage object 65. The storage
6 object, for example, is contained in a file or file system compatible with UNIX and MS-
7 Windows. Therefore, file access protocols such as NFS and CIFS may access the storage
8 object container file. Consequently, conventional facilities for data sharing and
9 protection may operate upon the storage object container file. Use of a file as a container
10 for the storage object may also exploit some file system features such as quotas, file
11 system cache in the data mover, and block allocation on demand.

12 The iSCSI protocol begins with a login process during which the iSCSI initiator
13 establishes a session with a target. TCP connections may be added and removed from a
14 session. The login process may include authentication of the initiator and the target. The
15 TCP connections are used for sending control messages, and SCSI commands,
16 parameters, and data.

17 FIG. 3 shows one type of an iSCSI PDU command 82. The command 82
18 includes a one-byte opcode indicating the command type, and two bytes of flags. The
19 first byte of flags includes two flags that indicate how to interpret the following length
20 field, and a flag set to indicate a read command. The second byte of flags includes one
21 Autosense flag and three task attribute flags. The command 82 includes a length
22 indicating the length of the command in bytes, and a Logical Unit Number (LUN)
23 specifying the Logical Unit to which the command is targeted. The command 82

1 includes an Initiator Task Tag assigned to each SCSI task initiated by the SCSI initiator.
2 A SCSI task is a linked set of SCSI commands. The Initiator Task Tag uniquely
3 identifies each SCSI task initiated by the SCSI initiator. The command 82 includes a
4 Command Reference Number (CMDRN) for sequencing the command, and an Expected
5 Status Reference Number (EXPSTATRN) for indicating that responses up to
6 EXPSTATRN-1 (mod 2^{32}) have been received. The command 82 includes an
7 Expected Data Transfer Length that the SCSI initiator expects will be sent for this SCSI
8 operation in SCSI data packets. The command 82 includes a 16-byte field 83 for a
9 Command Descriptor Block (CDB). The command 82 may also include additional
10 command-dependent data.

11 FIG. 4 shows a typical example of the SCSI Command Descriptor Block (CDB)
12 83 in the SCSI command of FIG. 3. In this example, the CDB 83 is ten bytes in length.
13 The CDB 83 includes a one-byte operation code, a four-byte logical block address (if
14 required), a two-byte parameter list (if required), and a control byte. SCSI disk drives
15 internally translate the logical block address to a physical cylinder, head, and sector
16 address in order to perform a read or write data access.

17 FIG. 5 shows a container file 84 for a storage object. The container file 84
18 includes conventional attributes 85 of the container file such as the type of file, the size of
19 the file, the date and time when the file was created, last modified, and last accessed, and
20 security information such as a list of users having access permissions and the access
21 permission of each user.

22 The conventional data storage area of the container file 84 includes a header 86 of
23 storage object metadata including storage object attributes, and a storage object data

1 storage area 87 for storage of data of the storage object. The storage object attributes 86
2 include a storage object type such as a disk drive or a logical volume of storage. The
3 logical volume of storage could be a raw volume, a sliced volume, a striped volume, or a
4 concatenated volume. A raw volume corresponds to one disk or one disk partition. A
5 sliced volume is partitioned, for example, into public and private regions. A striped
6 volume is striped across more than one disk. A concatenated volume results from the
7 combination of raw volumes, for example, to provide a volume size greater than the
8 maximum size of a basic raw volume.

9 The storage object attributes 86 also include a storage capacity in bytes, and the
10 amount of storage presently used, and the amount of free space in the storage object. The
11 storage object attributes 86 include a list of users permitted to access the storage object
12 through the SCSI termination module (64 in FIG. 2), and a respective permission and
13 quota for each user. Moreover, the storage object attributes may include configuration
14 information, such as a location (bus, target and LUN) of the storage object, and an
15 internal organization of the storage object, such as a level of redundancy in an array of
16 disk drives (RAID level) and a striping scheme. The specified internal organization of
17 the storage object could be used as a guide or specification for mapping of the data
18 storage area 87 of the container file 87 to storage in the cached disk array (49 in FIG. 2).

19 The various RAID levels include: no redundancy (level 0); mirrored disks (level
20 1); Hamming code for error correction (level 2); single check disk per group (level 3);
21 independent reads and writes (level 4); spread data/parity over all disks (no single check
22 disk)(level 5). These various RAID levels are described in Patterson et al., "A Case for
23 Redundant Arrays of Inexpensive Disks (RAID)," Report No. UCB/CSD 87/391,

1 Computer Science Division (EECS), University of California, Berkeley, California,
2 December 1987, pp. 1-24, incorporated herein by reference. Raid levels 2 to 5 imply
3 certain patterns of striping of data and check information across the disk array. The
4 striping pattern may affect access performance in various ways depending on whether the
5 access is read or write, or intermittent or continuous. For example, a striping pattern for
6 continuous media files is shown in FIG. 21 of Venkatesh et al., U.S. Patent 5,974,503
7 issued Oct. 26, 1999 and entitled "Storage and Access of Continuous Media Files
8 Indexed as Lists of RAID Stripe sets associated with file names," incorporated herein by
9 reference. Striping patterns for mirrored disks are disclosed in Venkatesh et al., U.S.
10 Patent 6,397,292 issued May 28, 2002, and entitled "Asymmetrical Striping of Mirrored
11 Storage Device Arrays and Concurrent Access to Even Tracks in the First Array and Odd
12 Tracks in the Second Array To Improve Data Access Performance," incorporated herein
13 by reference.

14 FIG. 6 shows a procedure for execution of a SCSI command by the SCSI
15 termination module in the data mover of FIG. 2. In a first step 91, execution continues to
16 step 92 if the SCSI command is a request for data access. In step 92, the capacity of the
17 storage object is obtained from the storage object attributes in the storage object container
18 file. In step 93, the logical block address specified in the SCSI command is compared to
19 the capacity in order to validate the address if the logical block address is less than the
20 capacity. If the logical block address is invalid, then execution branches from step 94 to
21 handle the error. Otherwise, execution continues to step 95. In step 95, the logical block
22 address is used to access the storage area of the storage object in the container file. In
23 particular, the logical block address from the SCSI command is added to the byte address

1 of the beginning of the storage object data in the storage object container file to provide
2 an address for accessing data in the container file.

3 In step 91, if the SCSI command is not a data access request, then execution
4 branches to step 96. In step 96, if the SCSI command is a request for storage attributes,
5 then execution continues to step 97. In step 97, the SCSI termination module accesses
6 storage attributes in the storage object container file. In step 98, the requested storage
7 attributes are returned to the SCSI device driver having originated the SCSI command.

8 If in step 96 the SCSI command is not a request for storage attributes, then
9 execution branches from step 96 to step 99, in order to execute the command.

10 In the data processing system of FIG. 2, it is desired to provide a snapshot copy
11 facility and an IP replication facility in the data mover 26. A snapshot copy provides a
12 point-in-time copy of the data saved on the storage object for on-line data backup
13 applications and data mining tasks. A snapshot copy facility also saves user disk space
14 by only storing the change in the user data since the last snapshot was taken. IP
15 replication can provide high availability of data by maintaining two or more replicas of
16 data on different network file servers at different sites that are geographically remote
17 from each other.

18 In the data processing system of FIG. 2, it is desired to permit the client 23 to
19 manage backup and replication of its SCSI storage object in the data mover 26 during
20 concurrent access to the storage object using the iSCSI protocol. For example, while the
21 client 23 writes data to the data mover 26, the data mover 26 replicates the data to the
22 second network file server 22 in FIG. 1 by transmitting a copy of the data over the IP
23 network 20 using the NFS or CIFS protocols. One way of doing this is to provide a

1 parallel and concurrent TCP connection between the client 23 and the data mover 26 for
2 control of snapshot copy and IP replication applications in the data mover 26. This
3 method is described below with reference to FIGS. 7 to 14.

4 As shown in FIG. 7, the client is provided with an application program called a
5 virtual block device manager 71 for managing backup and replication of the client's
6 storage object 65 in the data mover 26. In order to backup or replicate a consistent view
7 of the storage object 65, write access to the storage object by the SCSI device driver is
8 synchronized to the backup or replication process. For example, write access of the
9 storage object 65 is paused at the completion of a synchronous write, a commit operation
10 for a series of asynchronous writes, or a commit of a current transaction consisting of a
11 series of write operations. During the pause, a snapshot copy operation is initiated for the
12 backup or replication process.

13 One way of pausing write access to the storage object 65 at the completion of a
14 commit operation is to provide a service in the applications 51 or the file system 53 that
15 provides a notification to interested applications of the commit operation and suspends
16 further write operations to storage until an acknowledgement is received from the
17 interested applications. Although the Windows operating system 53 does not presently
18 provide such a service, the Microsoft Exchange application provides such a service.

19 In a MS Windows machine, the Windows Management Instrumentation (WMI)
20 facility 73 provides a mechanism for communication between processes. The WMI
21 facility 73 functions as a mailbox between processes in the client 23. A process may call
22 a WMI driver routine that places data into the WMI facility and notifies subscribers to the
23 data. In the example of FIG. 7, for example, the virtual block device manager 71 calls a

1 routine in a snapshot and replication dynamic link library (DLL) 72, which receives
2 notification of a commit event. For example, the Microsoft Exchange application
3 responds to an application program interface (API) call that invokes the service in
4 Exchange to suspend further write operations after a commit operation, and returns a
5 notification that further write operations have been suspended. A similar API is used in
6 UNIX file systems. This API call is provided in order to put the database such as
7 Exchange or Oracle in a quiescent state in order to make a backup copy of the database.
8 In the event of a system crash, the database application can replay its logs during
9 recovery to ensure that its backup database is brought back to a consistent state.
10 When a commit event has occurred and further writing over the iSCSI/TCP connection
11 (112 in FIG. 12) is inhibited, a network block services (NBS) driver 74 in the client
12 establishes a parallel and concurrent TCP connection (113 in FIG. 12) to a network block
13 services server 75 in the data mover (21 in FIGS. 11 and 12). NBS control commands
14 cause a snapshot copy facility 76 or an IP replication facility 77 to initiate a snapshot
15 copy or IP replication process upon the storage object 65. The snapshot copy or IP
16 replication process may continue as a background process concurrent with subsequent
17 write access on a priority basis when the SCSI termination 64 executes SCSI write
18 commands from the client's SCSI driver 54.

19 The IP replication facility may use the snapshot copy facility for the remote
20 replication process by transmission of data from the snapshot copies over the IP network
21 concurrent with client write access to the storage object 65. For example, the snapshot
22 copy facility periodically takes a snapshot of a consistent view of the storage object 65,

1 and the IP replication facility transmits the differences between the successive snapshots
2 over the IP network to a remote network file server.

3 The NBS protocol is introduced in Xiaoye Jiang et al., "Network Block Services
4 for Client Access of Network-Attached Data Storage in an IP Network," U.S Patent
5 Application Ser. 10/255,148 filed Sep. 25, 2002, incorporated herein by reference. This
6 protocol is extended for snapshot copy and replication of storage objects, as further
7 described below with reference to FIGS. 9 to 11. Details of a snapshot copy facility are
8 described in Keedem U.S. Patent 6,076,148 issued June 13, 2000, incorporated herein by
9 reference; and Philippe Armangau et al., "Data Storage System Having Meta Bit Maps
10 for Indicating Whether Data Blocks are Invalid in Snapshot Copies," U.S. Patent
11 Application Ser. 10/213,241 filed Aug. 6, 2002, incorporated herein by reference. Details
12 of an IP replication facility are described in Raman, et al., U.S. Patent Application Ser.
13 No. 10/147,751 filed May 16, 2002, entitled "Replication of Remote Copy Data for
14 Internet Protocol (IP) transmission," incorporated herein by reference; and Philippe
15 Armangau et al., Data Recovery With Internet Protocol Replication With or Without Full
16 Resync, U.S. Patent Application Ser No. _____ filed June 25, 2003, incorporated
17 herein by reference. The snapshot copy or IP replication facility, for example, operates
18 on a file system compatible with the UNIX and MS Windows operating systems. In this
19 case, the snapshot copy facility 76 or the IP replication facility 77 accesses the storage
20 object container file 84 through the UxFS file system 44 in the data mover 26.

21 The snapshot copy facility 76 may use a "write-anywhere" file versioning
22 method. A snapshot of a file initially includes only a copy of the inode of the original
23 file. Therefore the snapshot initially shares all of the data blocks as well as any indirect

1 blocks of the original file. When the original file is modified, new blocks are allocated
2 and linked to the original file inode to save the new data, and the original data blocks are
3 retained and linked to the snapshot inode. The result is that disk space is saved by only
4 saving the delta of two consecutive versions.

5 The IP replication facility 77 can be based on a snapshot copy facility 76 that
6 periodically saves the deltas between consecutive consistent versions of a file. In a
7 background process, the data mover transmits the deltas over the IP network to another
8 file server at a remote location. Upon confirmation of receipt of a delta at a remote
9 location, the data mover can delete its local copy of the delta.

10 The network block services driver 74 communicates with the network block
11 services server 75 using a relatively light-weight protocol designed to provide block level
12 remote access of network storage over TCP/IP. This protocol also provides remote
13 control of snapshot copy and IP replication facilities. The network block services server
14 75 maintains in memory a doubly-linked list of storage objects accessible to clients via
15 their network block services drivers. Each storage object is also linked to a list of any of
16 its snapshot copies. A copy of this list structure is maintained in storage. When the data
17 mover 26 reboots, the NBS server rebuilds the in-memory list structure from the on-disk
18 structure. The data mover 26 also maintains a directory of the storage objects using as
19 keys the file names of the storage object container files. The in-memory list structure and
20 the directory are extended to include the iSCSI storage objects, so that each iSCSI storage
21 object is accessible to a client through the SCSI termination 64 or the network block
22 services server 75. In particular, each virtual LUN recognized by the SCSI termination
23 64 has a corresponding NBS identifier recognized by the network block services server

1 75 and a corresponding storage object container file name. API calls are provided to
2 coordinate the iSCSI initiator 66 and the SCSI termination 64 with the NBS protocol
3 during snapshot operations. For example, the snapshot and replication DLL 72 includes
4 an API call through the WMI 73 to the iSCSI initiator 66 for changing the destination
5 address of the iSCSI protocol. This API call can be used during a restore operation, in
6 order to resume processing from a backup copy of the storage object 65 after a disruption.
7 The storage object 65 could be included in a storage object container file or could be a
8 raw volume of the storage array or any combination of volumes such as raw volumes,
9 slices, striped volumes or meta concatenated volumes. This approach has minimal
10 impact on upper layer components of the operating system of the client 23.

11 FIG. 9 shows an IP data packet encoded by the network block services driver (74
12 in FIG. 6). The data packet includes a packet header 80 and, when appropriate, data 81
13 appended to the packet header. The packet header, for example, has the following
14 format:

```
15  
16    struct PktHdr{  
17       unsigned long    OpCode;  
18       unsigned long    PduLen;  
19       unsigned long    PktId;  
20       RtnStat_t       Status;  
21       unsigned long    PktSeq;  
22       unsigned long    ConnGen
```

```

1      unsigned   Handle[MAX_NBS_HANDLE_LEN];

2      unsigned long   Reserved1

3      unsigned long   Reserved2;

4      unsigned long   DataLen

5      integer        Magic[MAGIC_LEN];

6      unsigned long   SubCmd;

7      unsigned long   Offset;

8      unsigned long   Padding[13];

9      unsigned long   CRC

10     };

```

11

12 These fields include an opcode field (OpCode), a packet data unit length field (PduLen),
13 a packet identifier field (PktId), a reply status field (Status), a packet sequence field
14 (PktSeq), a connection generation count field (ConnGen), an object handle field
15 (Handle), two reserved fields (Reserve1 and Reserve2), an offset field (Offset) for
16 specifying a start block offset, a data length field (DataLen), a magic field containing
17 “NBS” and a revision number, a sub command field (SubCmd), a padding field
18 (Padding), and a CRC field containing a cyclic redundancy check of the header excluding
19 the CRC field. The OpCode, PduLen, Status, Offset and DataLen fields in the packet
20 header are all represented as network byte order (i.e. big endian). All bits not defined
21 should be set to zero, and all reserved and padding fields should be set to zero as well.

FIG. 10 shows a table of some client opcodes in IP packets produced by the network block services driver (74 in FIG. 7) and transmitted from the client (23 in FIG. 7) to the network block services server (75 in FIG. 8). The client opcodes have the following format:

| | |
|--------|--------------|
| 0x0000 | READ |
| 0x0001 | WRITE |
| 0x0003 | INFO |
| 0x0004 | NO-OP |
| 0x0005 | PAUSE |
| 0x0006 | RESUME |
| 0x0007 | SNAP |
| 0x0008 | READ_OPAQUE |
| 0x0009 | WRITE_OPAQUE |
| 0x000a | AUTH |
| 0x000b | MSG |

A READ opcode is used when network block services driver requests the network block services server to read a specified length of data from a specified storage object beginning at a specified offset. A WRITE opcode is used when the network block

1 services driver requests the network block server to write a specified length of data to a
2 specified storage object beginning at a specified offset.

3 An INFO opcode is used when the network block services driver discovers
4 storage objects in the network block services server. It has two sub commands:
5 NBS_INFO_CMD_LIST and NBS_INFO_CMD_INFO.

6 NBS_INFO_CMD_LIST sub command is used to retrieve an NBS storage object
7 list on the server. NBS_INFO_CMD_INFO sub command is used to get the capacity and
8 handle information of a NBS storage object with a specific external name. The
9 parameters and results are encapsulated in XML format and attached to the packet
10 header.

11 For the LIST request, the client supplies authentication information, and the
12 server returns the list of storage object information to the client, including the external
13 name of the storage objects and their attributes. The attached XML format is defined as:

14

15 Request:

16 <nbsLstRqst/>

17

18 Reply:

19 <nbsLstRply>

20 <nbs name=\"%s\" blkSize=%lu numBlks=%Lu rw=%d share=%d

21 snapable=%d dr=%d tws=%d />

22 ...

23 </nbsLstRply>

1

2 For the INFO request, the client will provide the external name of the storage
3 object, the server will reply with the size of blocks and the total number of blocks for that
4 storage object.

5

6 Request:

7 <nbsInfoRqst nbsId=\"%s\" />

8

9 Reply:

10 <nbsInfoRply BlkSize=%lu NumBlks=%Lu rw=%d share=%d
11 xferSize=%u snapable=%d dr=%d tws=%d />

12

13

14 A “NO-OP” opcode is used when the network block services driver sends a
15 packet to the network block services server to get a return packet to test or keep alive a
16 TCP connection between the network block services driver and the network block
17 services server.

18 The PAUSE and RESUME commands are used to pause and resume access to a
19 specified storage object in order to ensure data consistency. For example, this is done
20 during system maintenance and snapshot operations.

21 The SNAP opcode is used for snapshot management. A sub command is included
22 for a specific snapshot operation such as create a snapshot of a storage object, delete a
23 snapshot, restore a storage object with a snapshot, refresh a snapshot, and list the
24 snapshots for a storage object.

1 The READ OPAQUE and WRITE OPAQUE permit a read or write of an opaque
2 data structure in a storage object.

3 The NBS driver uses the AUTH opcode to request a connection and provide
4 authentication to the NBS server. Upon receipt of a connection request, the NBS server
5 first checks an export list to decide whether to accept the connection. Once the
6 connection established, a one-way Challenge-Handshake Authentication Protocol
7 (CHAP) is performed to authenticate the NBS driver before accepting further NBS
8 commands from the NBS driver. The CHAP protocol includes the following steps:

9
10 1. The client sends a list of available authentication methods to the server. The
11 XML format is:

12
13 <nbsAuthMethodRqst>
14 <nbsAuthMethod name=\"%s\" />
15 ...
16 </nbsAuthMethodRqst>

17
18 2. The server sends back the authentication method reply with the method the
19 server chooses. The XML format is:

20
21 <nbsAuthMethodRply name=\"%s\" />
22

1 3. The client sends out algorithm code (CHAP_A) that it uses. The XML format
2 is:

3
4 <nbsAuthARqst CHAP_A=%d />

5
6 4. The server sends back a reply with identifier (CHAP_I) and the
7 challenge(CHAP_C). The XML format is:

8
9 <nbsAuthARply CHAP_A=%d CHAP_I=%x CHAP_C=\"%s\" />

10
11 5. The client sends the response (CHAP_R) back to the server. The CHAP_R is
12 calculated based on the secret mapped to the name (CHAP_N), CHAP_I, and CHAP_C.
13 The XML format is:

14
15 <nbsAuthRRqst CHAP_N=\"%s\" CHAP_R=\"%s\" />

16
17 6. If the CHAP_R calculated by the server is the same as the sent by the client,
18 the server sends back the reply indicating a successful authentication. The XML format
19 is:

20
21 <nbsAuthRRply />

22

1 If at any step the NBS driver fails to send out the correct request and data, then
2 the server would drop the connection. In this case, the NBS driver would need to restart
3 the connection and authentication process.

4 The MSG opcode is used to send a message from the NBS driver to the NBS
5 server. For example, messages could be sent to control an IP replication process. For
6 example, IP replication parameters would include a network name or IP network address
7 of a target file server to which the container file or container file system would be
8 replicated.

9 FIG. 11 shows the server opcodes used in IP data packets returned by the network
10 block services server to the network block services driver. A READ RETURN opcode is
11 used when the network block services server returns the data requested in a driver's read
12 request. The WRITE RETURN opcode is used when the network block services server
13 returns a confirmation of a write operation performed in response to a write request from
14 the network block services driver. The INFO RETURN opcode is used when the
15 network block services server returns information requested by the network disk client.
16 The NO-OP RETURN opcode is used when the network block services server returns a
17 NO-OP packet in response to a NO-OP packet from the network block services client. In
18 a similar fashion, the other return opcodes are used when the server returns requested
19 information or confirmation of receipt or execution of a corresponding command from
20 the NBS driver.

21 The server opcodes have the following format:

22

23 0x0040 READ RESPONSE

| | | |
|----|--------|-----------------------|
| 1 | 0x0041 | WRITE RESPONSE |
| 2 | 0x0043 | INFO RESPONSE |
| 3 | 0x0044 | NO-OP RESPONSE |
| 4 | 0x0005 | PAUSE RESPONSE |
| 5 | 0x0006 | RESUME RESPONSE |
| 6 | 0x0007 | SNAP RESPONSE |
| 7 | 0x0008 | READ_OPAQUE RESPONSE |
| 8 | 0x0009 | WRITE_OPAQUE RESPONSE |
| 9 | 0x000a | AUTH RESPONSE |
| 10 | 0x000b | MSG RESPONSE |

11

12 In the packet header (100 in FIG. 9), the “PduLen” field indicates the total length
13 of packet header 100 and data 101. In INFO and NO-OP operations, the “PduLen” field
14 is set to the length of the Packet Header. In a WRITE request operation or a READ
15 reply, the “PduLen” field is set to the length of the Packet Header and Data Segments. In
16 READ request operation or WRITE reply, the “PduLen” field is represented as the length
17 of Packet Header.

18 In the packet header (100 in FIG. 9), the “PktId” field is a unique identifier of the
19 packet. The “PktId” field is set by the driver, and need not be changed by the server.

20 In the packet header (100 in FIG. 9), the “Status” field is zeroed out by the driver,
21 and the server sets up and returns status depending on the success of the requested
22 operation. For example, the server returns an indication of whether or not the requested

1 operation succeeds or fails. For a failed operation, a specific error code may be returned,
2 for example, indicating that a specification is invalid, no memory is available, an object
3 to be accessed is busy or frozen, or a CRC error has occurred. For receipt of a corrupted
4 data packet, a time-out for a response to a request, or for many other failures,
5 retransmission of a request from the driver may be appropriate. If a failure persists after
6 retransmission, then the driver will attempt to connect to the next data mover in the
7 network file server of the NBS server. The NBS driver maintains an outstanding request
8 queue in order to reissue the outstanding requests during this recovery process.

9 In the packet header (100 in FIG. 9), the “PktSeq” field contains a sequence
10 number of the request packets. Due to network failure or server fail-over, the NBS
11 packets may be lost during transmission between the driver and the server. Sometimes,
12 the packets should be resent. However, some of the NBS requests such as SNAP
13 requests are non-idempotent, and resending those requests can cause incorrect
14 configuration of the storage object if the server responds to duplicate requests. The
15 PktSeq number is used to ensure that the server does not respond to duplicate requests.

16 In the packet header (100 in FIG. 9), the “ConnGen” field contains a generation
17 count of the client side connection for a particular storage object. The ConnGen field is
18 used by a Linux NBS client to keep track of resend and fail over activities, and to
19 invalidate orphan packets.

20 In the packet header (100 in FIG. 9), the “Handle” field contains an object handle.
21 The object handle is a sixteen bytes array that contains a connection handle used to
22 identify the storage objects and connection instance for each request.

1 In the packet header (100 in FIG. 9), the “Reserve1” and “Reserve2” fields are
2 reserved for future use.

3 In the packet header (100 in FIG. 9), the “Offset” field is the offset of the volume,
4 and it is a count of a number of blocks in the logical volume. For example, each block
5 consists of 8 K bytes. The Offset is only meaningful for READ and WRITE operations.

6 In the packet header (100 in FIG. 9), for a read request, the “DataLen” field
7 specifies the number of bytes in a Data segment 81 following the packet header 80. For a
8 read request, the “DataLen” field specifies the number of bytes to be read from the
9 specified volume, starting at the Offset into the volume.

10 In the packet header (100 in FIG. 9), the “Magic” field identifies the version of
11 the NBS driver, in order to permit downward compatibility in case of future
12 enhancements.

13 In the packet header (100 in FIG. 9), the “SubCmd” field contains the sub-
14 command for the INFO and SNAP commands.

15 FIG. 12 shows the control flow through the client and server of FIGS. 7 and 8 for
16 processing snapshot and replication requests from a system administrator 100. This
17 control flow results from operation of the virtual block device manager 71 in FIG. 12 in
18 accordance with the flowchart in FIGS. 13 and 14.

19 In a first step 121 of FIG. 13, the virtual block device manager receives a
20 snapshot or replication request from the system administrator or another application
21 program of the client. In step 122, the virtual block device manager invokes the DLL
22 routine for a snapshot or replication of the virtual block device. In step 123, the call of
23 the routine in the Windows operating system, or a kernel call in the UNIX operating

1 system, for a snapshot or replication of the virtual block device initiates a sync and
2 suspend iSCSI application interface (API) call to WMI 73. This call is relayed to the
3 Exchange application (111 in FIG. 12). Similar calls would be relayed to other
4 applications using virtual block devices to be snapshotted or replicated. Then in step 124
5 the virtual block device manager sets a timer and then suspends its execution, until
6 execution is resumed by receiving a callback notification that Exchange or other
7 applications have completed a sync and suspend operation, or by expiration of the timer.
8 In step 125, if execution has been resumed but no callback was received, then an error is
9 logged indicating that the Exchange application has failed to perform the sync and
10 suspend iSCSI operation within the timer interval. Otherwise, if a callback has been
11 received, then execution continues to step 126. In step 126, the virtual block device
12 manager sends a snapshot or replicate command to the data mover via the NBS TCP
13 connection. After step 126, execution continues in step 127 of FIG. 14.

14 In step 127 of FIG. 14, the virtual block device manager sets a timer and suspends
15 execution. Execution is resumed upon a callback from the network block services driver
16 reporting that a snapshot or replication has been initiated, or upon expiration of the timer
17 interval. In step 128, if execution has been resumed but no callback was received, then
18 an error is logged indicating that the data mover has failed to initiate a snapshot or
19 replication within the timer interval. If a callback was received, then execution continues
20 to step 129. In step 129, the DLL for snapshot or replication initiates resumption of the
21 iSCSI operation by the Exchange or other applications.

22 Although a storage object such as a virtual disk drive or volume could be
23 contained in a single file as shown in FIG. 5, it is also possible to contain the storage

1 object in a file system. As shown in FIG. 15, such a file system includes a storage object
2 file system directory providing directory entries for a storage object attribute file 132, a
3 storage object data file 133, and a storage object log file 134. The data area of the storage
4 object data file 133, for example, would contain the storage object attributes, and the data
5 area of the storage object data file 133 would contain the data of the storage object. The
6 file system may also include a storage object log file 134, which could be used by a client
7 owning the storage object for any purpose, such as a log of the history of access to the
8 storage object. The use of such a file system instead of a single file to contain a storage
9 object would be advantageous in a file server that does not provide range locking within a
10 file. In this case, file-locking contention would be reduced between the storage object
11 attribute file and the storage object data file. Also, the storage object data file 133 would
12 have the advantage that logical block address in the SCSI command block could directly
13 address the storage object data file for read and write operations.

14 Although the use of the SCSI and NBS protocols have been described above with
15 respect to clients and file servers in an IP network, it should be understood that the SCSI
16 and NBS protocols could be used in other kinds of networks, such as Ethernet,
17 Asynchronous Transfer Mode (ATM), or Fibre-Channel (FC) networks. For example,
18 the SCSI or NBS commands could be encapsulated in the data packets of the Ethernet,
19 ATM, or FC networks. It would also be possible to use the FC protocol over a FC
20 network for block level access of a client to a storage object in the server in lieu of a
21 SCSI protocol.

22 In view of the above, there has been described a method of containing a storage
23 object such as a virtual disk drive or storage volume in a file in order to provide access to

1 the storage object by a low-level protocol such as SCSI, iSCSI, or FC concurrent with
2 access to the container file by a high-level protocol such as NFS or CIFS. This permits
3 block level access via different types of network connections such as SAN and NAS
4 concurrent with file system sharing by clients with diverse operating systems, and fast
5 file system backup, fail-over, and recovery.

6

7